Analysing Effect of Database Grouping on Multi-Database Mining

Animesh Adhikari, Lakhmi C. Jain, Sheela Ramanna

Abstract — **In many applications we need to synthesize global patterns in multiple large databases, where the applications are independent of the characteristics of local patterns. Pipelined feedback technique (PFT) seems to be the most effective technique under the approach of local pattern analysis (LPA). The goal of this paper is to analyse the effect of database grouping on multi-database mining. For this purpose we design a database grouping algorithm. We introduce an approach of non-local pattern analysis (NLPA) by combining database grouping algorithm and pipelined feedback technique for multi-database mining. We propose to judge the effectiveness of non-local pattern analysis for multi-database mining. We conduct experiments on both real and synthetic databases. Experimental results show that the approach to non-local pattern analysis does not always improve the accuracy of mining global patterns in multiple databases.**

Index Terms — **Local pattern analysis, Multi-database mining, Non-local pattern analysis, Pipelined feedback technique, Synthesis of patterns**

I. Introduction

MULTI-database mining is strategically an essential area of data mining. This is because of the fact that in many applications we need to process data from various sources [12], [13], [4], [8]. As a result, research in multi-database mining is gaining momentum [18], [20], [6].

In many situations data are collected from different regions across the globe. It might be possible to move data from one place to another place for some applications that are independent of the local properties of databases. The goal of this paper is to judge whether one could improve mining global patterns by sacrificing local properties of patterns in multi-databases. In an earlier work [7], we have shown that PFT improves the quality of global patterns significantly as compared to an existing technique [15], [17], [19], [5] that scans each database only once. In an effort to make further improvements, we introduce non-local pattern analysis for multi-database mining and propose to study its effectiveness in synthesizing global patterns. There are two primary reasons for non-local pattern analysis (i) the local properties of patterns need not always to be preserved; (ii) the number of estimations of a pattern might get decreased.

Local pattern analysis [19], [5] is an important approach of mining multiple large databases. One could obtain reasonably good solutions for a large class of problems. In local pattern analysis, each local database is mined locally. Then every branch forwards the local pattern base to the central location. All the pattern bases are then processed for synthesizing global patterns in multiple databases. It is important to observe that the same pattern might not get reported from every local database. As a result, the local pattern analysis is an approximate method of mining multiple large databases. If we are able to amalgamate all the databases together then there is no difference between mono-database mining and multi-database mining. There might be different reasons in different contexts that prohibit us to amalgamate all the databases together [6]. The next question comes to our mind is that whether one could reduce the frequency of database mining. In this regard, there are two extreme cases of multi-database mining viz., mono-database mining and local pattern analysis. Mono-database mining is used when there is a possibility of clubbing all the local databases. But the latter is used when each local database requires mining locally. In the first case, the frequency of mining database is one. But the frequency of mining is equal to the number of local databases in case of local pattern analysis. In view of reducing the frequency of mining, one may need to group the databases and then each group of databases is mined separately. Moreover, when we group some databases, the databases in a group are mined together. Thus, the number of estimations of a pattern will be reduced. For the purpose of constructing groups we consider that the groups of databases are mutually exclusive and exhaustive. The mutually exclusiveness property ensures that a database belongs to only one of the different groups. On the other hand exhaustiveness property ensures that each database belongs to a group. We club all the databases in a group for the purpose of multi-database mining. In this arrangement one needs to estimate a pattern less number of times, but local properties of a pattern may not get restored. This may have a bearing on the quality of the global patterns. In this paper, we investigate whether such an arrangement of local databases enhances accuracy of the global patterns.

Grouping of databases seems to an important issue for discovering knowledge in multiple databases. Wu and Zhang.

Animesh Adhikari is with the Department of Computer Science, S P Chowgule College, Goa, India (phone: 91-0832-2759504; fax: 91-0832-2759067; e-mail: animeshadhikari@yahoo.com).

Lakhmi C. Jain is with the School of Electrical and Information Engineering, University of South Australia, Mawson Lakes Campus, Australia (e-mail: Lakhmi.Jain@unisa.edu.au).

Sheela Ramanna is with the Department of Applied Computer Science, University of Winnipeg, Winnipeg, Canada (e-mail: s.ramanna@uwinnipeg.ca).

[16] have proposed a similarity measure sim_1 to identify similar databases based on item similarity. The authors have designed a clustering algorithm based on measure sim_1 to cluster databases for the purpose of selecting relevant databases. Such clustering is useful when the similarity is based on items in different databases. Item similarity measure sim_1 might not be useful in many multi-database mining applications where clustering of databases is based on some other criteria. For example, if we are interested in the databases based on transaction similarity then the above measures might not be appropriate. We have designed an algorithm for database clustering based on transaction similarity [4]. For this purpose, we have proposed a similarity measure $simi_1$ to cluster databases. One could group some objects based on an external criterion also. For example, the available main memory could pose a constraint in multi-database mining. It might be difficult to mine all the databases together when the databases are large. We will discuss later how the available main memory induces database grouping for the purpose of multi-database mining.

In an earlier work [7], we performed many experiments using different multi-database mining techniques (MDMTs). Experimental results have shown that PFT outperforms each of the existing techniques that scans a database only once. We introduce an approach of non-local pattern analysis based on PFT. For the purpose of completeness we present PFT in Section III.

Data mining applications based on multiple databases could be broadly categorized into two groups. The applications in the first group are based on patterns in individual databases. On the other hand, the second group of applications deals with the global patterns in multiple databases that are distributed in different geographical regions. Our goal is to study the effectiveness of non-local pattern analysis for mining global patterns in multiple databases. In many applications one may not have any restriction on moving a local database from one branch to another branch. Therefore, one could amalgamate a few branch databases and then mine a group of databases together. Then another group of databases could be formed and mined together, and so on. Finally, one could synthesize global patterns from the patterns in these groups of databases. We propose to study the effect of such grouping on synthesizing global patterns.

Rest of the paper is organized as follows. In Section II, we discuss related work. We present pipelined feedback technique in Section III. In Section IV, we introduce a non-local pattern analysis. We present a heuristic-based grouping algorithm in support of non-local pattern analysis. A discussion on finding the best grouping can be found in Section V. We present experimental results in Section VI.

II. Related Work

Zhang et al. [17] have proposed algorithm *IdentifyExPattern* for identifying global exceptional patterns in multi-databases. Here every local database is mined separately at random order using mono-database mining technique for synthesizing global exceptional patterns. As a result, the synthesized global patterns might deviate significantly from the true global patterns. We have proposed an algorithm *Association-Rule-Synthesis* [5] for synthesizing association rules in multiple real databases. This algorithm is useful for real databases, where the trend of the customers' behaviour exhibited in one database is usually present in other databases. For synthesizing high frequency association rules, Wu and Zhang [15] have proposed *RuleSynthesizing* algorithm for synthesizing high frequency association rules in multiple databases. Based on the association rules in different databases, the authors have estimated weights of different databases. Let w_i be the weight of the i-th database, $i = 1, 2, \ldots, n$. Without any loss of generality, let the association rule r be extracted from the first m databases, for $1 \le m \le n$. Actual support of r in D_i, $supp_a(r, D_i)$, has been assumed as 0, for $i = m + 1, m + 2, \ldots, n$. Then the support of r in D has been synthesized as follows.

$$supp_s(r, D) = w_1 \times supp_a(r, D_1) + \ldots + w_m \times supp_a(r, D_m) \quad (1)$$

This method is an indirect approach and computationally expensive as compared to other techniques. Existing parallel mining techniques [2], [9] could also be used to deal with multiple large databases. In the context of pattern synthesis, Viswanath et al. [14] have proposed a novel pattern synthesizing method called *partition based pattern synthesis* which can generate an artificial training set of exponential order when compared with that of the given original training set.

III. Pipelined Feedback Technique (PFT)

For the purpose of completeness, we first present an overview of PFT [7]. Consider a multi-branch organization that collects data from multiple local branches. Let D_i be the database corresponding to the i-th branch, $i = 1, 2, \ldots, n$. Also let LPB_i be the local pattern base for D_i, $i = 1, 2, \ldots, n$. Also, let D be the union of all branch databases.

Let $D_1, D_2, \ldots, D_n$ be an arrangement of mining databases. First D_1 is mined using a mono-database mining technique [3], [11], and local pattern base LPB_1 is extracted. While mining D_2, all the patterns in LPB_1 are extracted irrespective of their values of interestingness measures such as minimum support and minimum confidence. Apart from these patterns, some new patterns that satisfy user-defined thresholds of interestingness are also extracted. In general, while mining D_i all the patterns in D_{i-1} are extracted irrespective of their values of interestingness, and some new patterns that satisfy user-defined thresholds of interestingness are also extracted. Due to this nature of mining each database, the technique is called a feedback model. Thus, $|LPB_{i-1}| \le |LPB_i|$, $i = 2, 3, \ldots, n$. There are $n!$ arrangements of pipelining for n databases. All the arrangements of databases might not produce the same mining result. If the number of local patterns increases, we get more accurate global patterns and a better analysis of local patterns. An arrangement of local databases would produce near optimal result if $|LPB_n|$ is maximal. Let $size(D_i)$ be the size of D_i (in bytes), $i = 1, 2, \ldots, n$. We shall follow the following rule of thumb regarding the arrangements of databases for the purpose of mining: The number of patterns in D_{i-1} is greater than or equal to the number of patterns in D_i, if $size(D_{i-1}) \ge size(D_i)$, $i = 2, 3, \ldots, n$. For the

purpose of increasing number of local patterns, D_{i-1} precedes D_i in the pipelined arrangement of mining databases if $size(D_{i-1}) \geq size(D_i)$, $i = 2, 3, \ldots, n$. Finally, we analyze the patterns in LPB_1, LPB_2, …, LPB_n for synthesizing global patterns, or analyzing local patterns.

Most of the databases are sparse. A pattern might not get reported from all the databases. However, once a pattern gets mined from a database, it also gets reported from the remaining databases in the pipeline. Thus, PFT improves the accuracy of multi-database mining. In the Section IV, we shall introduce an approach of non-local pattern analysis and we analyse its effectiveness in Section VI.

For synthesizing global patterns in D we discuss here a simple pattern synthesizing (SPS) algorithm with the help of itemset pattern in a database. Without any loss of generality, let the itemset X be extracted from the first m ($\leq n$) databases. Then synthesized support of X in D could be obtained as follows:

$$supp_s(X, D) = \frac{1}{\sum_{i=1}^{n} |D_i|} \times \sum_{i=1}^{m} [\, supp_a(X, D_i) \times |D_i| \,] \quad (2)$$

The accuracy of global pattern X increase as m approaches to n. The concepts of accuracy and error of a pattern are opposite to each other. When the error of a pattern increases, we say that its accuracy decreases, and vice-versa. We explain the concept of error in the following section.

A. Error

Let $D_1, D_2, \ldots, D_n$ be n branch databases. Also, let $size(D_1) \geq size(D_2) \geq \ldots \geq size(D_n)$. In PFT, the databases are mined according to the following order: $D_1, D_2, \ldots, D_n$. An itemset X gets reported from some of the given databases. In PFT, once X is reported from one of the given databases, then it also gets mined from the remaining databases. Suppose X is reported first time from D_k at minimum support level α, for $1 \leq k \leq n$. Then the error of mining X in D could be expressed as follows:

$$\text{Error}(X, D) = |supp_a(X, D) - supp_e(X, D)| \quad (3)$$

where, $supp_a(X, D)$ and $supp_e(X, D)$ denote the actual (apriori) support [3] and the estimated support of X in D, respectively. The supports $supp_a(X, \bigcup_{i=k+1}^{n} D_i)$ and $supp_e(X, \bigcup_{i=k+1}^{n} D_i)$ are the same, since X gets reported from the databases D_{k+1}, D_{k+2}, …, and D_n at minimum support level α. Thus, the error of mining X in D could be expressed as follows:

$$\text{Error}(X, D) = \left| supp_a(X, \bigcup_{i=1}^{k} D_i) - supp_e(X, \bigcup_{i=1}^{k} D_i) \right| \quad (4)$$

As the value of k increases, the amount of error increases provided the method of estimating support remains the same. Therefore, if the itemset X gets mined early in the pipelined arrangement, then amount of error decreases. In other words, as the number of estimations reduces, the error of mining itemset X reduces. This is an important observation and has been applied to the proposed non-local pattern analysis. Let S be the set of all itemsets synthesized from D. Then the average error (AE) of the experiment could be defined as follows:

$$\text{AE} = \frac{1}{|S|} \sum_{X \in S} Error(X, D) \quad (5)$$

Also, one could define maximum error (ME) of the experiment as follows:

$$\text{ME} = maximum\ \{Error\ (X, D) |\ X \in S\} \quad (6)$$

IV. Non-Local Pattern Analysis (NLPA)

Consider a multi-branch organization that has n (≥ 2) branches. Suppose that each branch maintains a database of all local transactions. The goal of this paper is to investigate whether one could improve multi-database mining by sacrificing local properties of the patterns. In view of this one could group databases induced by available main memory. Let k be the number of groups of databases. Different groups of databases are given as follows: $\{D_{11}, D_{12}, \ldots, D_{1n_1}\}$, $\{D_{21}, D_{22}, \ldots, D_{2n_2}\}$, …, $\{D_{k1}, D_{k2}, \ldots, D_{kn_k}\}$, where $D_{ij} \in \{D_1, D_2, \ldots, D_n\}$, for $j = 1, 2, \ldots, n_i$; $i = 1, 2, \ldots, k$; $\sum_{i=1}^{k} n_i = n$; $n_i \geq 1$. Afterwards each group of databases are amalgamated and mined. The crux of non-local pattern analysis is how to group the databases so that one could mine each group of databases effectively within the limited memory. We formulate the problem of grouping databases as follows.

A. Grouping Databases

Multi-database mining could be performed by amalgamating some local databases and mining them together. But the performance of data mining process seems to be constrained by size of the main memory. If the available main memory is less, it might take a longer time to accomplish the mining task. During the grouping process, we shall continue to club databases as long as main memory is available. Let β be the optimum size of available main memory. Let $size(D)$ be the size of database D. Then the problem of grouping databases can be stated as follows:

We are given a set of numbers $S = \{size(D_1), size(D_2), \ldots, size(D_n)\}$. *Our objective is to find r subgroups* $S_1, S_2, \ldots S_r$, *for some* $r \geq 1$, *so that* $\sum_{i=1}^{r-1} \left(\beta - \sum_{j=1}^{n_i} size\left(D_{ij}\right) \right)$ *is a minimal, where the following conditions are true.*

(*i*) $\sum_{j=1}^{n_i} size(D_{ij}) \leq \beta$, *for* $i = 1, 2, \ldots, r$, *and* $D_{ij} \in \{D_1, D_2, \ldots, D_n\}$

(*ii*) $S_i = \{size(D_{i1}), size(D_{i2}), \ldots, size(D_{in_i})\}$, *and* $S_i \subseteq S$, $i = 1, 2, \ldots, r$

(*iii*) $S_i \bigcap S_j = \phi$, $\forall\ i \neq j$, *and* $\bigcup_{i=1}^{r} S_i = S$

$\beta - \sum_{j=1}^{n_i} size\left(D_{ij}\right)$ is the amount of *unutilized space* for the i-th group, $i = 1, 2, \ldots, r$. The goal of the grouping process is to reduce the total amount of unutilized spaces. In the next section, we propose a heuristic algorithm that utilizes main memory effectively.

B. An Heuristic Algorithm for Grouping Databases

As the number of groups decreases, one needs to estimate a global pattern fewer number of times. If the number of groups is one then all the patterns are exact and become true representative of the multiple databases. Given a limited amount of memory, it is important to group the databases so that it can fit best in the main memory. During the grouping

process, if the larger databases are not considered at the early stage of grouping, then it could pose problems. As a result, the number of groups might increase. Smaller databases can be accommodated in a group easily, since their sizes are small. We apply this heuristic to design a grouping algorithm. Let us take an example to illustrate the grouping process.

Example 1. Let N be the set of sizes of the given databases. Let N be {139, 29, 43, 152, 165, 74, 5, 120}. Also let β be 200. First we sort the numbers in N in non-decreasing order. The ordered numbers are given as follows: 5, 29, 43, 74, 120, 139, 152, 165. The maximum size among the given databases is 165 bytes. First, we form a group with the database of size 165 bytes. Otherwise, it might cause producing a larger amount of unutilized space. Then along with the database of size 165 bytes, we club the database of size 29 bytes so that their sum 194 still remains less than or equal to 200. The database of size 29 bytes is obtained by searching the list from the right hand side. Any database of size in between 29 bytes and 165 bytes can not be clubbed with the database of size 165 bytes, since their sum would exceed 200 bytes. No more databases can be clubbed with them. Otherwise, their sum could exceed the available memory. In this case, the available memory is 200 bytes. As a result, the first group G_1 = {165, 29} is formed with an unutilized space of 6 bytes. Now we consider the database of size 152 bytes, since it is the second maximum among the given database sizes. Proceeding in the same way, one could form the second group as G_2 = {152, 43, 5} with an unutilized space of 0 byte. Then the next group G_3 = {139} is formed with unutilized space of size 61 bytes. The final group is G_4 = {120, 74} with unutilized space of 6 bytes. The total amount of unutilized spaces is equal to (6 + 0 + 61 + 6) bytes i.e., 73 bytes. Such grouping of databases might not be unique. For example, there exists another grouping of databases viz., {{152, 43, 5}, {165}, {139, 29}, {120, 74}}, that results in the same amount of unutilized spaces. •

Lemma 1. Let β be the optimum size of available main memory. Also, let D_{ij} be a database in group G_i, for j = 1, 2, …, n_i and i = 1, 2, …, r. Then the following grouping results in the same amount of unutilized spaces, provided $|G_j| + |D_{ik}| \leq \beta$:
$G_1, G_2, \ldots, G_{i-1}, G_i - \{D_{ik}\}, G_{i+1}, \ldots, G_{j-1}, G_j \bigcup \{D_{ik}\}, G_{j+1}, \ldots, G_r$, for some $i \neq j$. •

Based on the procedure illustrated in Example 1, we present here a heuristic algorithm, *Database-Grouping*, as follows.

procedure *Database-Grouping* (n, A, β)
Input:
n: number of databases
A: array of database sizes
β: maximum available memory (in bytes)
Output:
k: number of groups
G: two dimensional array representing different groups
01: sort A in non-decreasing order;
02: **let** k = 0;
03: **for** i = 1 to n **do**
04: *allocation*(i) = 0;
05: **end for**
06: **let** *index* = n;
07: **while** (*index* ≥ 1) **do**
08: **let** i = *index*; **let** *sum* = 0; **let** *col* = 1;
09: increment k by 1;
10: **while** (*sum* ≤ β) and ($i \geq 1$) **do**
11: **if** (*sum* + $A(i) \leq \beta$) and (*allocation*(i) = 0) **then**
12: *sum* = *sum* + $A(i)$; *allocation*(i) = 1;
13: increment *col* by 1; $G(k, col) = A(i)$;
14: **end if**
15: decrease i by 1;
16: **end while**
17: $G(k, 1) = col$-1;
18: **let** $j = n$;
19: **while** (allocation(j) ≠ 0) and ($j \geq 1$) **do**
20: decrement j by 1;
21: **end while**
22: **let** *index* = j;
23: **end while**
24: **for** i = 1 to k **do**
25: display the members of the i-th group;
26: **end for**
end procedure

We explain here the different variables and parts of the above algorithm. The number of groups is returned through the variable k. Here G is a two dimensional matrix that stores the output groups. The i-th row of G stores the i-th output group, i = 1, 2, …, k. The first element of each row contains the number of elements in that group as noted in line 17. The subsequent elements are the database sizes in that group. The databases, whose sizes are kept in a group, are required to be clubbed for the purpose of mining. Initially, all the databases are unallocated (lines 03-05), since there exists no group. The database having a maximal size is allocated first in a group. Therefore, *index* variable gets initialized to n (line 06). The inner *while-loop* constructs a group of databases that are amalgamated afterwards for the purpose mining (lines 10-16). When a database is included in a group, the corresponding allocation tag is changed to 1 (line 12). Lines 18-22 help finding the next position (*index*) in the array A from which we start allocating the element for the next group. All the elements at the right side of current value of *index* are allocated to different groups.

Algorithm *Database-Grouping* forms k groups from the given databases, for some $k \leq n$. Once the groups are formed, then we amalgamate the databases in each group for the purpose of mining. Accordingly, we have k amalgamated databases. We then follow pipelined feedback technique for mining these k databases.

The accuracy of synthesized patterns would depend on the sizes of the databases. In PFT, we mine first the database having the maximum size. It is expected that the database having the largest size would produce the maximum number of patterns. Further, PFT extracts all the previously extracted patterns irrespective of their interestingness values. Thus, it is always better to mine the largest database right at the beginning. The procedure *Database-Grouping* helps maximizing the database at every step by clubbing the databases. Moreover, it

applies a heuristic approach while forming a group of databases.

In the context of mining time-stamped databases [8], *Database-Grouping* algorithm might play an important role. The time granularity of time-stamped databases is an important issue. Again, the time granularity would depend on an application. If time granularity is smaller, for example a month, then each of the monthly databases is expected to smaller. The procedure *Database-Grouping* would produce better grouping of databases. As a result, *Database-Grouping* algorithm is expected to produce good grouping when the size of each database is small.

Lemma 2. Let n be number of databases and k be the number of groups returned by the *Database-Grouping* algorithm. The time complexity of the algorithm is $O(n \times k)$.

Proof. *For-loop* in lines 3-5 takes $O(n)$ time. The algorithm returns k groups. Therefore, the outer *while-loop* in lines 7-23 repeats k times. The inner *while-loop* in lines 10-16 could repeat $O(n)$ times for each iteration of outer *while-loop*. Also, the *while-loop* in lines 19-21 could repeat n times for an iteration of outer *while-loop*. In lines 26-28, we display all the members in every group. Therefore, it takes $O(n)$ time. Thus, the time complexity of the algorithm is *maximum* $\{O(n), O(n \times k), O(n)\}$, i.e., $O(n \times k)$ time. •

C. Accuracy of mined patterns

If all the branch databases are amalgamated and mined then there is no difference between multi-database mining and mono-database mining. In this case a reported pattern is 100% accurate. But such situation may not exist always. Many branch databases could be very large. As a result the data mining process could consume unreasonable amount of time. In some cases it might not be possible to complete the data mining task. As a result a multi-database mining technique might report approximate patterns. An approximate pattern is not true representative pattern in multiple databases.

In our earlier work [7], we have noted that the accuracy of a mined pattern using PFT is generally higher than that of any other existing technique. This is true because of the fact that once a pattern is reported from a branch database, it also gets reported from the databases mined afterwards. If we can increase the size of each group (G_i) as much as possible by amalgamating branch some databases (D_j), the experimental results have shown that the average accuracy of a mined pattern might not decrease, for $i = 1, 2, \ldots, r$; $j = 1, 2, \ldots, n$. As we increase the size of each G_i, we expect more patterns to be generated at each stage. Specifically, if a large number of patterns are reported at the initial stages of mining then the accuracy of those patterns, when synthesized globally, become higher. This is because of the fact that if a pattern is reported at any stage then it also gets reported subsequently due the application of feedback mechanism. Let us consider those patterns that are reported at the latter stages. These patterns might differ significantly from the actual global patterns. Therefore, the error of the experiment, AE and / or ME, might be more for non-local pattern analysis that that of PFT.

It might be appealing if one attaches depth of data mining with a mined pattern. We define *depth* of a pattern in multi-database mining as the fraction of total sizes of group databases from which a pattern gets extracted to the total size of all databases. Let $G_1, G_2, \ldots, G_r$ be the group of databases mined sequentially. Let pattern p be reported first time from the k-th group i.e., G_k. If p is an itemset pattern, then one could report its *depth* along with its support [1]. Thus,

$$depth\ (p) = (|G_k| + |G_{k+1}| + \ldots + |G_r|) / |D|,$$

where $(|G_1| + |G_2| + \ldots + |G_r|) = |D|$, $0 < depth\ (p) \leq 1$. Depth of a pattern represents the amount of data from which it has been extracted from a multi-database environment. If the depth of p is 1, then it is exact. One could discard a pattern if its depth is low.

V. An Optimal Grouping of Databases

Let us refer to algorithm *Database-Grouping* presented in Section IV. In most of the cases, it produces good grouping of databases. But it may not result in an optimal grouping for the purpose of multi-database mining. One could determine all possible groupings of databases at a given a set of databases and β. Then one could find the amount of unutilized spaces for every grouping. In the worst case one needs $O(n^2)$ comparisons to form a group, where n is the number of databases. Thus, the worst case complexity of such optimal algorithm is $O(n^2 \times k)$, where k is the number of groups. Such an algorithm might not be always attractive when a simpler algorithm like *Database-Grouping* produces an optimal result in the most of cases.

VI. Experimental Results

We have carried out several experiments to study the proposed approach of mining global patterns in multiple large databases. All the experiments have been implemented on a 2.8 GHz Pentium D dual core processor with 988 MB of memory using visual C++ (version 6.0) software. We present experimental results using synthetic dataset *T10I4D100K* [10] and two real datasets *retail* [10] and *BMS-Web-Wiew-1* [10]. We present some characteristics of these datasets in Table I. Let *NT*, *AFI*, *ALT*, and *NI* be the number of transactions, average frequency of an item, average length of a transaction, and number of items in a database, respectively. Each of the above datasets is divided into 10 databases for purpose of conducting our experiments.

TABLE I
DATASET CHARACTERISTICS

Dataset	*NT*	*ALT*	*AFT*	*NI*
T10I4D100K	1,00,000	11.10	1276.12	870
retail	88,162	11.31	99.67	10,000
BMS-Web-Wiew-1	1,49,639	2.00	155.71	1,922

TABLE II
T10I4D100K DATABASE CHARACTERISTICS

DB	NT	*size*(DB)	ALT	AFI	NI
T0	1,000	40	11.09	12.70	795
T1	2,000	81	11.18	24.43	834
T2	3,000	119	11.01	35.45	847
T3	4,000	159	11.01	46.84	855
T4	8,000	323	11.15	93.79	866
T5	10,000	400	11.05	115.93	867
T6	12,000	483	11.12	140.28	866
T7	15,000	605	11.13	175.24	867
T8	20,000	807	11.14	233.31	869
T9	25,000	1,027	11.07	290.38	867

TABLE III
retail DATABASE CHARACTERISTICS

DB	NT	*size*(DB)	ALT	AFI	NI
R0	1,000	36	9.52	5.11	1,000
R1	2,000	96	11.91	11.57	830
R2	3,000	143	11.72	16.22	862
R3	4,000	181	11.17	20.15	873
R4	8,000	358	11.10	4015	899
R5	10,000	473	11.49	49.82	1,097
R6	12,000	565	11.33	55.91	1,218
R7	14,000	634	10.76	58.34	1,311
R8	16,000	744	11.04	65.80	1,389
R9	18,162	922	11.89	77.87	1,500

TABLE IV
BMS-Web-Wiew-1 DATABASE CHARACTERISTICS

DB	NT	*size*(DB)	ALT	AFI	NT
B0	1,000	10	2.0	5.13	195
B1	2,000	22	2.0	3.56	157
B2	3,000	35	2.0	5.01	77
B3	5,000	63	2.0	3.05	1637
B4	10,000	131	2.0	6.23	1605
B5	15,000	205	2.0	1500	10
B6	20,000	273	2.0	2000	10
B7	25,000	341	2.0	2500	10
B8	30,000	410	2.0	3000	10
B9	38,639	528	2.0	3863.8	10

We have generated these databases arbitrarily consisting of a good mix of small and large databases. The databases obtained from *T10I4D100K*, *retail* and *BMS-Web-Wiew-1* are named as *Ti*, *Ri*, and *Bi* respectively, for $i = 0, 1, \ldots, 9$ and subsequently referred to as input databases. Some characteristics of these input databases are presented in Tables II, III, and IV. Let N_T be {40, 81, 119, 159, 323, 400, 483, 605, 807, 1,027}, the set of sizes of databases obtained from *T10I4D100K*. Let N_R be {36, 96, 143, 181, 358, 473, 565, 634, 744, 922}, the set of sizes of databases obtained from *retail*. Also, let N_B be {10, 22, 35, 63, 131, 205, 273, 341, 410, 528}, the set of sizes of databases obtained from *BMS-Web-Wiew-1*. In Table V, we present some outputs showing that the proposed non-local pattern analysis does not always improve accuracy of patterns in multiple large databases.

TABLE V
ERROR OF THE EXPERIMENTS AT A GIVEN MINIMUM SUPPORT

Dataset	*T10I4D100K*	*retail*	*BMS-Web-Wiew-1*
Minimum support	0.045	0.15	0.075
Error type	AE	AE	AE
MDMT: PFT + SPS	0.00451	0.00478	0.00206
MDMT: NLPA	0.00452	0.00499	0.00333
Error type	ME	ME	ME
MDMT: PFT + SPS	0.02411	0.01191	0.00702
MDMT: NLPA	0.02418	0.01270	0.00781

We apply *Database-Grouping* algorithm presented above. The choice of β for each of the three databases is an important issue. The sizes of *T10I4D100K*, *retail* and *BMS-Web-Wiew-1* are 3.83 MB, 3.97 MB, 1.97 MB respectively. Therefore, it might be possible to fit all the 10 databases in main memory for conducting experiments using each of the three datasets. But for the purpose of applying *Database-Grouping* algorithm one could consider β as little more than the maximum size of the generated databases, and accordingly, we taken β as 1,100 KB, 1,000 KB, and 700 KB for conducting experiments using datasets *T10I4D100K*, *retail* and *BMS-Web-Wiew-1*, respectively. The groups formed for above three datasets are given below:

Group corresponding to *T10I4D100K*, G_T = {{1027, 40}, {807, 159, 119}, {605, 483}, {400, 323, 81}} with the total amount of unutilized space is equal to (33 + 15 + 12 + 296) bytes i.e., 356 bytes.
Group corresponding to *retail*, G_R = {{922, 36}, {744, 181}, {634, 358}, {565, 143, 96}, {473}} with the total amount of unutilized space is equal to (42 + 75 + 8 + 196 + 527) bytes i.e., 848 bytes.
Group corresponding to *BMS-Web-Wiew-1*, G_B = {{528, 131, 35}, {410, 273, 10}, {341, 205, 63, 22}} with the total amount of unutilized space is equal to (6 + 7 + 69) bytes i.e., 82 bytes.

Now we club the databases in each group for purpose of mining multi-databases. Let the databases generated for the first, second and third groups be D_{Ti}, $i = 1, 2, 3, 4$; D_{Rj}, $j = 1,2, 3, 4, 5$; and D_{Tk}, $k = 1, 2, 3$, respectively. We present the databases after grouping in Tables VI, VII and VIII.

TABLE VI
NEW DATABASES GENERATED FROM *T10I4D100K*

Generated databases	Databases to be clubbed
D_{T1}	T_9, T_0
D_{T2}	T_8, T_3, T_2
D_{T3}	T_7, T_6
D_{T4}	T_5, T_4, T_1

TABLE VII
NEW DATABASES GENERATED FROM *RETAIL*

Generated databases	Databases to be clubbed
D_{R1}	R_9, R_0
D_{R2}	R_8, R_3
D_{R3}	R_7, R_4,
D_{R4}	R_6, R_2, R_1
D_{R5}	R_5

TABLE VIII
NEW DATABASES GENERATED FROM *BMS-WEB-VIEW-I*

Generated databases	Databases to be clubbed
D_{B1}	B_9, B_4, B_2
D_{B2}	B_8, B_6, B_0
D_{B3}	B_7, B_5, B_3, B_1

We have conducted experiments on the new databases by applying PFT and non-local pattern analysis. In Figs. 1, 2, and 3, we have presented results of AE with respect to minimum supports. Experimental results show that PFT reports more accurate global patterns than non-local pattern analysis in the most of the cases. Also, we observe that there no fixed trend of AE over the increased support values.

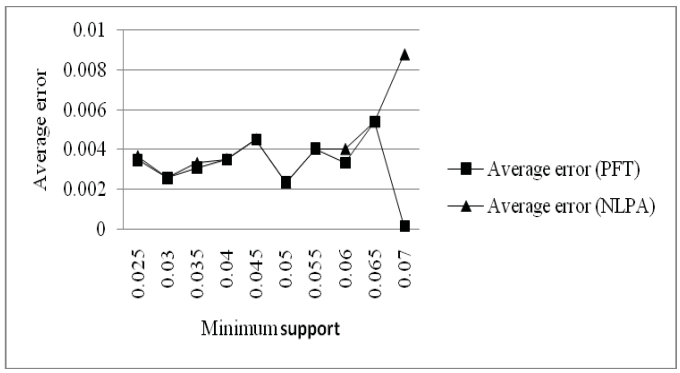

Fig. 1. Average error versus minimum support (for *T10I4D100K*)

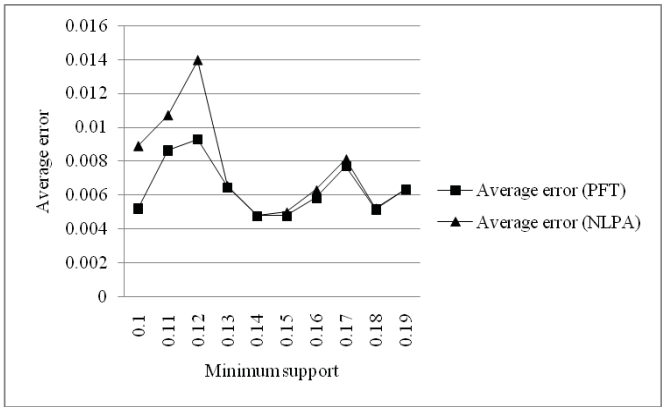

Fig. 2. Average error versus minimum support (for *retail*)

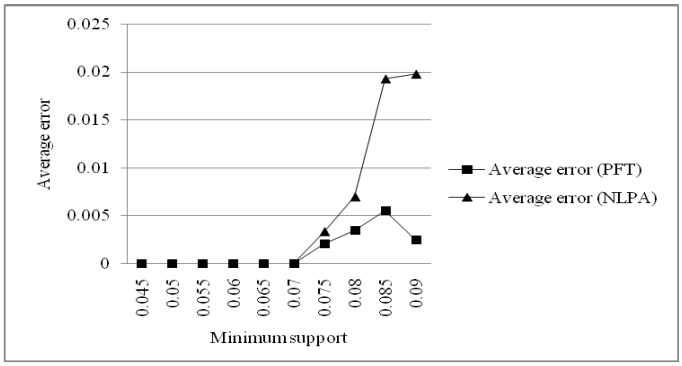

Fig. 3. Average error versus minimum support (for *BMS-Web-Wiew-1*)

VII. CONCLUSION

In this paper we have introduced non-local pattern analysis for multi-database mining in an attempt to study its effectiveness in synthesizing global patterns. A database grouping algorithm induced by main memory constraint has been introduced to applying non-local pattern analysis. Main memory constraint is an illustration of a criterion used for database grouping. Apparently non-local pattern analysis looks to be attractive, since the frequency of data mining is less as compared to local pattern analysis. As a result one needs to estimate a pattern lesser number of times for the purpose of synthesizing the global pattern. The drawback of non-local pattern analysis is that the patterns reported only from the last few groups might contribute significantly to the error of the experiment. This is due to the fact that a pattern is assumed absent when it does not get reported. Therefore, a mined pattern needs to be associated with the amount of data that it represents. For this purpose we have defined depth of a pattern in multi-database mining. A pattern becomes useless if its depth is low. We have conducted several experiments on real and synthetic datasets. Experimental results show that non-local pattern analysis might not be a better technique than PFT.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," In *Proceedings of ACM SIGMOD Conf. Management of Data*, 1993, pp. 207-216.

[2]R. Agrawal, and J. Shafer, "Parallel mining of association rules," *IEEE Transactions on Knowledge and Data Engneering*, vol. 8, no. 6, pp. 962-969, 1999.

[3] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," In *Proceedings of VLDB*, 1994, pp. 487-499.

[4] A. Adhikari, and P. R. Rao, "Efficient clustering of databases induced by local patterns," *Decision Support Systems*, vol. 44, no. 4, pp. 925-943, 2008.

[5] A. Adhikari, and P. R. Rao, "Synthesizing heavy association rules from different real data sources," *Pattern Recognition Letters*, vol. 29, no. 1, pp. 59-71, 2008.

[6] A. Adhikari, P. R. Rao, W. Pedrycz, *Developing multi-database mining applications*, Springer, 2010.

[7] A. Adhikari, P. R. Rao, B. Prasad, and J. Adhikari, "Mining multiple large data sources," *International Arab Journal of Information Technology*, vol. 7, no. 2, pp. 243-251, 2010.

[8] J. Adhikari, P. R. Rao, and A. Adhikari, "Clustering items in different data sources induced by stability," *International Arab Journal of Information Technology*, vol. 6, no. 4, pp. 394-402, 2009.
[9] J. Chattratichat, J. Darlington, M. Ghanem, Y. Guo, H. Hüning, M. Köhler, J. Sutiwaraphun, H.W. To, and D. Yang, "Large scale data mining: Challenges, and responses," In *Proceedings of KDD*, 1997, pp. 143-146.
[10] Frequent Itemset Mining Dataset Repository, http://fimi.cs.helsinki.fi/data/.
[11] J. Han, J. Pei, and Y. Yiwen, "Mining frequent patterns without candidate generation," In *Proceedings of SIGMOD*, 2000, pp. 1-12.
[12] D. Page, and M. Craven, "Biological applications of multi-relational data mining," *SIGKDD Explorations* vol. 5, no. 1, pp. 69-79, 2003.
[13] W. –C. Peng, Z. –X. Liao, "Mining sequential patterns across multiple sequence databases," *Data & Knowledge Engineering*, vol. 68, no. 10, pp. 1014-1033, 2009.
[14] P. Viswanath, M.N. Murty, and S. Bhatnagar, 2006. "Partition based pattern synthesis technique with efficient algorithms for nearest neighbor classification," *Pattern Recognition Letters*, vol. 27, no. 14, pp. 1714-1724, 2006.
[15] X. Wu, and S. Zhang, "Synthesizing high-frequency rules from different data sources," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 353-367, 2003.
[16] X. Wu, C. Zhang, and S. Zhang, "Database classification for multi-database mining," *Information Systems*, vol. 30, no. 1, pp. 71-88, 2005.
[17] C. Zhang, M. Liu, W. Nie, and S. Zhang, "Identifying global exceptional patterns in multi-database mining," *IEEE Computational Intelligence Bulletin*, vol 3, no 1, pp. 19-24, 2004.
[18] S. Zhang, C. Zhang, X. Wu, *Knowledge discovery in multiple databases*. Springer, 2004.
[19] S. Zhang, X. Wu, C. Zhang, "Multi-database mining," *IEEE Computational Intelligence Bulletin*, vol. 2, no. 1, pp. 5-13, 2003.
[20] S. Zhang, and M. J. Zaki, "Mining multiple data sources: Local pattern analysis," *Data Mining and Knowledge Discovery* (*Special issue*), Springer, 2006.